UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/720,614	11/24/2003	Martin G. Rammel	03-0945	4243

74576 7590 07/20/2010

HUGH P. GORTLER
23 Arrivo Drive
Mission Viejo, CA 92692

EXAMINER	
DAO, THUY CHAN	
ART UNIT	PAPER NUMBER
2192	
MAIL DATE	DELIVERY MODE
07/20/2010	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application Number: 10/720,614
Filing Date: November 24, 2003
Appellant(s): RAMMEL, MARTIN G.

Hugh P. Gortler, Esq.
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed April 22, 2010 appealing from the Office action mailed January 13, 2010.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

6,883,147	BALLAGH et al.	04-2005
2002/0103839 A1	OZAWA et al.	08-2002

Definitions cited from website Answers.com	Doc Code: NPL, 3 pages	01-13-2010
Admitted Prior Art (APA)	Specification, pages 1-2	

(9) Grounds of Rejection

The following grounds of rejection are applicable to the appealed claims:

❑ **Claim 31 is rejected under 35 U.S.C. 102(e) as being anticipated by Ballagh (US Patent No. 6,883,147).**

Claim 31:

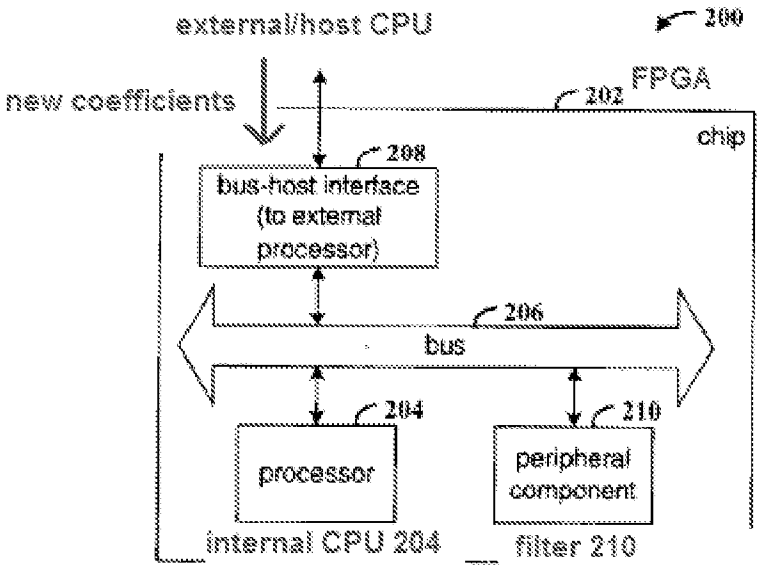
Ballagh discloses *a method of performing a numerical simulation with a Field Programmable Gate Array (FPGA) and a separate central processing unit (CPU), the method comprising:*

using the CPU (e.g., FIG. 2, an "external processor" (host CPU) coupled to bus-host interface 208, col.5: 22-28)

to perform a numerical simulation including generating input signals (e.g., col.4: 47-64, the host CPU and a system-level simulation environment 110) *and*

a FPGA (e.g., FIG. 2, chip 202, which is "an FPGA from Xilinx", col.5: 6-9)

sending the input signals to the FPGA (e.g., col.5: 28, from the external processor (host computer/CPU), transferring new filter coefficients to processor 204 (embedded within chip/FPGA 202))*;*



annotated **FIG. 2**

using the FPGA to apply a model to the input signals (e.g., FIG. 2, col.5: 6-14, chip/FPGA 202 has peripheral component 210, which includes a reconfigurable digital filler to process the input signals, col.5: 19-22; FIG. 3A, col.5: 39-54, said reconfigurable digital filter has specific "control logic that manages coefficient reloading, adjusts data rates, and controls filter output frame buffering", i.e., applying a specific model to the input signals) *and send results of the model back to the CPU* (e.g., col.5: 29-34)

the FPGA also generating a first output that marks data as valid or invalid (e.g., FIG. 3A, FPGA 202 generating "coef_we" (first output) to mark "coef" valid or invalid and sending "coef_we" to its sub-component FIR filter 250, col.5: 55-65),

a second output that indicates the first sample of each frame (e.g., FIG. 3A, output port "yn" indicating data in output frames, col.5: 29-38 and 43-48), *and*

a third output that indicates when the model can accept data (e.g., FIG. 3A, FPGA 202 generating "rfd" indicating status "busy" or not, col.5: 55-65)*; and*

wherein the CPU uses the results in the numerical simulation and the outputs to maintain data flow with the FPGA (e.g., col.6: 30-39, when the FIFO is full,

initiating a filter reload sequence and issuing read requests to obtain new coefficients from the external processor, col.5: 34-38).

❑ **Claims 32-37 are rejected under 35 U.S.C. 103(a) as being unpatentable over Ballagh in view of APA (Admitted Prior Art).**

Claim 32:

Ballagh does not explicitly disclose *the method of claim 31, wherein the input signals include sine wave functions representing real and imaginary inputs; and wherein the model includes a FFT.*

However, in an analogous art, APA further discloses *the input signals include sine wave functions representing real and imaginary inputs; and wherein the model includes a FFT* (e.g., page 2: 4-23).

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine APA's teaching into Ballagh's teaching. One would have been motivated to do so to transform a digitized waveform in the time domain into a digital representation in the frequency domain using Simulink simulation software as suggested by APA (e.g., page 2: 5-9 and 20-23).

Claim 33:

Ballagh discloses *the method of claim 32, wherein the FPGA converts inputs from double point precision to fixed point prior to performing the transform* (e.g., col.3: 52-65; col.5: 29-38)*; and wherein the FPGA converts the results from fixed point back to double precision prior to sending the results back to the CPU* (e.g., col.4: 65 – col.5: 38; col.6: 47-65).

APA further discloses *the input signals include sine wave functions representing real and imaginary inputs; and wherein the model includes a FFT* (e.g., page 2: 4-23).

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine APA's teaching into Ballagh's teaching. One would have been motivated to do so to transform a digitized waveform in the time domain into

a digital representation in the frequency domain using Simulink simulation software as suggested by APA (e.g., page 2: 5-9 and 20-23).

Claim 34:

APA further discloses *the method of claim 32, wherein the CPU performs a numerical simulation of a radar system* (e.g., page 2: 4-15).

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine APA's teaching into Ballagh's teaching. One would have been motivated to do so to as set forth above.

Claim 35:

Ballagh discloses *an apparatus* which recite(s) the same limitations as those of claim 31, wherein all claimed limitations have been addressed and/or set forth above. Therefore, as the reference teaches all of the limitations of the above claim(s), it also teaches all of the limitations of claim 35.

APA further discloses *a numerical simulation of sine wave functions representing real and imaginary inputs; and performing an FFT on the inputs* (e.g., page 2: 4-23).

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine APA's teaching into Ballagh's teaching. One would have been motivated to do so to transform a digitized waveform in the time domain into a digital representation in the frequency domain using Simulink simulation software as suggested by APA (e.g., page 2: 5-9 and 20-23).

Claim 36:

Ballagh discloses *the apparatus of claim 35, wherein the FPGA converts the real and imaginary inputs from double point precision to fixed point prior to performing the transform* (e.g., col.5: 1-38; col.6: 47-65)*; and*

wherein the FPGA converts the results of the FFT from fixed point back to double precision prior to sending the results back to the CPU (e.g., col.3: 52-65; col.4: 65 – col.5: 38).

Claim 37:

APA further discloses *the apparatus of claim 35, wherein the CPU performs a numerical simulation of a radar system* (e.g., page 2: 4-15).

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine APA's teaching into Ballagh's teaching. One would have been motivated to do so to as set forth above.

❑ **Claim 17 is rejected under 35 U.S.C. 103(a) as being unpatentable over Ballagh in view of APA and Ozawa (US Patent Publication No. 2002/0103839 A1).**

Claim 17:

The rejection of claim 32 is incorporated. Ballagh discloses *coupling an output of a double delay block to a third input of the FFT block, the third input being adapted to mark data input as valid or invalid* (e.g., FIG. 3A, FPGA 202 generating "coef_we" (first output) to mark "coef" valid or invalid and sending "coef_we" to its sub-component FIR filter 250, col.5: 55-65).

APA further discloses *performing receiving the real and imaginary inputs at first and second inputs of an FFT block via a pair of gateway in blocks* (e.g., page 2: 4-23).

Neither Ballagh nor APA explicitly discloses other limitations. However, in an analogous art, Ozawa discloses:

coupling an output of a k=0 block to a fourth input of the FFT block (e.g., [0166] and [1203]),

the fourth input being adapted to control a forward or a reverse transform (e.g., [0351], performing cascade processing signals);

coupling outputs of FFT block to at least one D flip flop-based registers adapted to provide a signal latency; and coupling the outputs of the registers to at least one gateway out (e.g., [1268]-[1269]).

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine Ozawa's teaching into Ballagh and APA's teaching. One would have been motivated to do so to process data in an arithmetic device as suggested by Ozawa (e.g., [0007]-[0011]).

-o-o-o-

(10) Response to Argument

I. Rejection of claim 31 under USC 102(e) as being anticipated by Ballagh US Patent No. 6,883,147 (Brief, pp. 8-10)

a) Limitation at issue "*numerical simulation*" (Brief, pp. 8-9):

Appellant stated that examiner's interpretation is not consistent with the specification (page 8, first portion).

Examiner respectfully disagrees. The claim merely calls for "*A method of performing a numerical simulation with a Field Programmable Gate Array (FPGA) and a separate central processing unit (CPU)…*" (claim 31, lines 1-3, emphasis added) and similarly recited in claim 35, lines 1-3, i.e., using a FPGA and a CPU to perform a numerical simulation for any purpose/design.

In response to Appellant's argument that the references fail to show certain features of Appellant's invention, it is noted that the features upon which Appellant relies (i.e., "… *numerical simulations are used to model different types of physical phenomena ... can be used to predict electromagnetic scattering... a specific example: a radar simulation*", Brief, page 8, first portion, emphasis added) are not recited in the rejected claim(s). Although the claims are interpreted in light of the specification, limitations from the specification are not read into the claims. See *In re Van Geuns*, 988 F.2d 1181, 26 USPQ2d 1057 (Fed. Cir. 1993).

Ballagh teaches:

"using the CPU to perform a numerical simulation" (e.g., simulating system parameters such as sample rates, data precision as arithmetic values using floating point and/or fixed point and simulating said floating point and/or fixed point, i.e., "*numerical simulation*" as claimed). For example:

> "The modeling phase consists of capturing the design (physical system) in an executable form, simulating, then

> analyzing the results. The modeling phase is appropriate for algorithm exploration, in which system parameters such as sample rates, data precision, and choice of functional blocks are decided. This process is iterative, with the results of analysis leading to revisions that allow system specifications to be met (design system performance). In the modeling phase, a high level of abstraction is desirable in order to facilitate algorithm exploration. For example, it is common to represent arithmetic values using floating point or fixed point rather than as buses of logic signals. Sampled data systems are also most conveniently modeled by defining sample rates rather than using explicit interconnections ("wires") representing clock and associated control signals (e.g., enable, reset)." (col.1: 32-46, emphasis added).

As previously addressed/presented in the previous Office action mailed January 13, 2010 (pages 2-3), well-known definitions in the art define:

floating-point

(flō'ting-point')
adj.
Of, relating to, or being a method of writing numeric quantities with a mantissa representing the value of the digits and a characteristic indicating the power of the number base, such as 3×10^{-5}.

fixed-point

(fikst'point')
adj.
Of, relating to, or being a method of writing numerical quantities with a predetermined number of digits and with the decimal located at a single unchanging position.

binary point

('bīn·ə·rē 'point)

(*computer science*) The character, or the location of an implied symbol, that separates the integral part of a numerical expression from its fractional part in binary notation.

Examiner further notes that Appellant's disclosure discussed,

> "More specifically, in the field of radar, numerical simulations of radar receivers may be used to predict radar performance versus various targets. A common algorithm used in these simulations is the Fast Fourier Transform (FFT) which transforms a digitized waveform in the time domain into a digital representation in the frequency domain… The method 10 is representative of at least some conventional methods for simulating radar signal processing using , one or more of the methods embodied in the SIMULINK simulation software developed by The Mathworks, Inc. of Natick, Mass." (specification, page 2, lines 4-23, numerical simulation can be performed using Simulink software product, emphasis added).

Similarly to said discussion, Ballagh discloses:

> "In a specific example, system 100 illustrates how a Simulink.RTM. system model is transformed by a MATLAB function (netlister) into an internal representation. This internal representation undergoes several transformations that resolve system parameters into the required control circuitry and target library mapping. In particular, data types are resolved into hardware-realizable forms, and clock signals, flip-flop clock enables, and resets are inferred from system sample rates. A

> processor design object will typically provide an integer-based data path. The system level translation automatically resolves fixed-point data into the underlying integer-based microprocessor instructions." (col.4: 11-23, Simulink system model includes simulating system parameters including integers, fixed-point data, i.e., *"numerical simulation"* as claimed, emphasis added).

Accordingly, as noted above, Ballagh provides a means for modeling a design system (physical) and simulating the system parameters/arithmetic values represented by floating point or fixed point (numerical input formats) so that allow system specifications (design system performance) to be met.

That is to say Ballagh teaches and/or provides a simulation means for modeling and/or predicting for such a design system performance (physical phenomena), and that is very much inlined with what Appellant's specification (page 2, lines 4-6, "More specifically, in the field of radar, numerical simulations of radar receivers may be used to predict radar performance versus various targets", emphasis added).

Thus, in view of the plain language of the claim ("numerical simulation") and well-known definitions in the art, "*a numerical simulation*" does not exclude simulating numeric quantities and/or arithmetic values represented by a numerical input format such as integers, floating-point, fixed-point, and binary-point formats (emphasis added).

b) Appellant further argued (page 8, second portion),

> Moreover, the interpretation of claim 31 is not consistent with the interpretation that those skilled in the art would reach, as required by MPEP 2111. Ballagh states that arithmetic values can be represented by fixed or floating point values during a numerical simulation (col. 1, lines 39-42). Ballagh does not state that a fixed or floating point operation is a numerical simulation.

Examiner respectfully disagrees. As set forth above, Ballagh discloses a system-level simulation environment 110 Simulink (col.4: 11-14) simulates system parameters/arithmetic values represented by floating-point or fixed-point (col.1: 32-46) and/or integers (col.4: 20-23), i.e., *"numerical simulation"* as claimed.

Contrast with Appellant's arguments, it is also well-known in the art that such a numerical simulation has been implemented with a numerical input format as fixed or floating point operations.

For example, see US Patent No. 5,963,731 to Sagawa et al., provided hereto under Attachment 1, pages 1-3, at the end of this Examiner's Answer, where Sagawa teaches that,

> "The present invention relates to a method of assisting execution of a plurality of numerical simulation programs for simulating physical phenomena on a computer so that the programs are executed in cooperation with each other." (col.1: 6-9, emphasis added); and
>
> "Then, the resource quantity of each computer is obtained. The resource quantity includes a CPU power, a main memory quantity, a disk I/O power, and a network I/O power. To be specific, the CPU power denotes an arithmetic capability of the processor, or an arithmetic operation quantity executable in a unit time. In the simulation program used in the present embodiment, floating-point operations are mainly performed, so that FLOPS (FLoating-point Operation Per Second) is used for the unit for the CPU power…" (col.34: 16-24, emphasis added).

c) Appellant' arguments regarding "*speeding up a numerical simulation*" (Brief, page 9).

In response to Appellant's argument that the references fail to show certain features of Appellant's invention, it is noted that the features upon which Appellant relies (i.e., "*speeding up a numerical simulation*", emphasis added) are not recited in the rejected claim(s). Although the claims are interpreted in light of the specification, limitations from the specification are not read into the claims. See In re Van Geuns, 988 F.2d 1181, 26 USPQ2d 1057 (Fed. Cir. 1993).

Furthermore, examiner notes that in light of the plain language of the claim, claim 31, lines 1-3, "*speeding up a numerical simulation*" at most would include using both a CPU and a FPGA to perform the numerical simulation.

As illustrated in FIG.2, Ballagh discloses an "external processor" (a host CPU external to a FPGA chip 202) and the FPGA chip 202 for performing numerical simulation.

It should be noted that the FPGA 202 is being used as a co-processing/co-simulating unit with the "external processor" (host CPU) – and that would arguably be what so-called "*speeding up*" the numerical simulation – see further details under subsections d) and f) below (regarding "*a portion of a simulation being offloaded from a processor to an FPGA*" and "*send results of the model back to the CPU*", respectively).

d) Appellant further argued, "*Ballagh is silent about a portion of a simulation being offloaded from a processor to an FPGA, … Ballagh only describes a simulation for designing an FPGA, wherein the entire simulation is run on a single computer.*" (Brief, page 9, emphasis added).

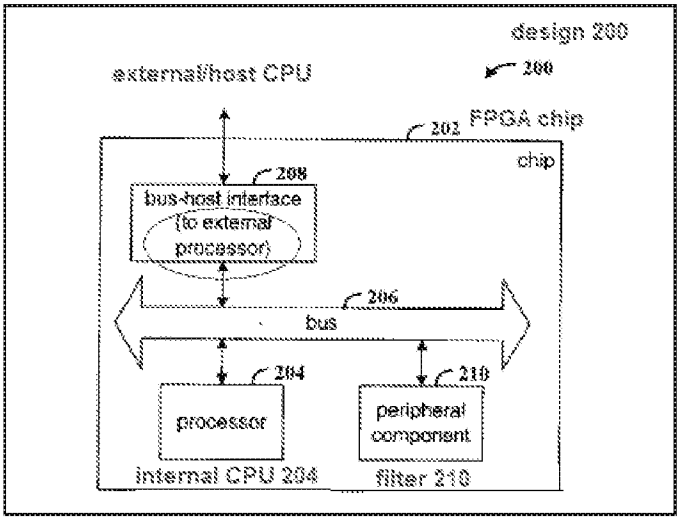
As an initial matter, examiner notes that the plain language of the claim does not require the numerical simulation is performed on more than one computer - *See* also subsection c) about "speeding up" as addressed above.

Although Ballagh teaches a single (host) computer, said single (host) computer has both a FPGA and an "external processor" (a host CPU external to the FPGA) as claimed (FIG.2, col.4: 65 – col.5: 28).

For example, in FIG.2, Ballagh teaches an "external processor" (a host CPU external to the FPGA) transfers ("offloaded") new coefficients (input signals) to FPGA 202 (i.e., "*a portion of a simulation being offloaded from a processor to an FPGA*" as claimed) via bus interface 208, and then said FPGA 202 processes (simulates) the received coefficients ("the input signals" as claimed):

> "A hardware realization of design 200 operates in two modes: filter reloading and filter frame data transfer. When the filter (i.e., the peripheral component) is not being reloaded, frames of filter output are transferred over the bus to (internal) processor 204. The frames are then sent (sent back) from the (internal) processor on bus 206 to a host computer ("external processor"/host CPU, external to FPGA 202) for analysis. On the host (external/host CPU), the user may construct a new filter and transfer new coefficients to the processor (internal processor 204) via bus interface 208 and bus 206..." (FIG.2, col.5: 29-38, emphasis added).

host computer (col.5: 22-28) includes:
host CPU (external to FPGA chip 202) and
FPGA chip 202 (having internal CPU 204)



annotated **FIG. 2**

It should be also noted that by offloading the new coefficients (input signals) to the FPGA 202, arguably the "external processor" (host CPU) provides what so-called "*a portion of a simulation being offloaded from a processor to an FPGA"* as claimed and that it would also provide what so-called "speeding up a numerical simulation" of the external/host CPU. And Ballagh would do so as simulated results from the FPGA are being sent back to the external/host CPU – *See* more detail in subsection f) below.

e) Appellant further argued, "*Ballagh does not describe the use of a CPU to perform a numerical simulation including generating input signals and sending the input signals to the FPGA. Ballagh only describes an external processor for generating filter coefficients, and an on-chip processor 204 for loading the coefficients into a FIR filter*." (Brief, page 9).

Examiner notes that, as acknowledged above by Appellant, "...*Ballagh only describes an external processor for generating filter coefficients* (generating "input signals" as claimed)*, and an on-chip processor 204 for loading the coefficients* (after the on-chip processor 204 receiving the coefficients from the sending external/host processor, now the on-chip processor 204 sending/loading the received coefficients to the FIR filter) *into a FIR filter."* (emphasis added).

Please note that on-chip processor 204 and FIR filter (i.e., the peripheral component 210 – see col.5: 29-31) are both internal to the FPGA chip 202. That is to say, the coefficients have been generated and sent from the "external processor" (the host CPU external to the FPGA) to the FPGA chip 202,

> "In an example application, the host computer (the external processor/host CPU) initiates filter reloading and transfers new filter coefficients to processor 204 (internal processor 204 in FPGA 202). Upon receiving new coefficients from the host, the processor (204) controls the filter reloading

> from within the FPGA." (FIG.2, col.5: 24-28, emphasis added).

Examiner notes that the claimed limitations "*generating input signals and sending the input signals to the FPGA*" does not exclude the "external processor" (the host CPU external to the FPGA) for generating filter coefficients ("*input signals*" as claimed), and sending the filter coefficients ("*the input signals*" as claimed) to FPGA chip 202 (including internal/on-chip processor 204 and filter 210) as illustrated in FIG.2 and col.5: 29-54.

f) Appellant's arguments in Brief, page 9, last paragraph, seems to direct to particular claimed limitations "*using the FPGA to apply a model to the input signals and send results of the model back to the CPU"* (claim 31, lines 6-7).

Examiner respectfully disagrees with Appellant's arguments. Ballagh teaches:

"*using the FPGA to apply a model to the input signals"* (e.g., FIG. 2, col.5: 6-22, FPGA chip 202 has peripheral component 210, which includes a reconfigurable digital filler (col.5: 29-31) to process (simulate) the input signals, i.e., applying the reconfigurable digital filter ("apply a model") to process/filter the coefficients ("the input signals"), and

FIG. 3A, col.5: 39-54, details of said reconfigurable digital filter ("a model"):

> "FIG. 3A illustrates an example FIR filter logic block 250 available to a designer as an abstract, user-selectable library element in a system such as Sysgen. Tools such as Sysgen work well in modeling high performance custom signal processing data paths. To illustrate how to extend a data path into a peripheral component, the reconfigurable filter 250 is used as an example. The library element that defines the filter includes control logic that manages coefficient reloading, adjusts data rates, and controls

> filter output frame buffering." (i.e., apply the reconfigurable filter ("apply a model" as claimed) to the coefficients ("input signals" as claimed) to manage reloading, adjust rates, and control buffering, emphasis added);

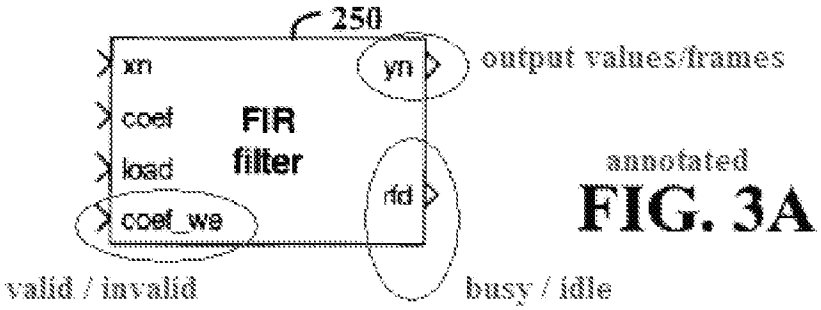
"and send results of the model back to the CPU"

> "A hardware realization of design 200 operates in two modes: filter reloading and filter frame data transfer. When the filter (i.e., the peripheral component) is not being reloaded, frames of filter output are transferred over the bus to (internal) processor 204. The frames are then sent (sent back) from the processor (internal processor 204) on bus 206 to a host computer ("external processor"/host CPU) for analysis." (col.5: 29-24, sending frames processed by the FPGA back to external processor/host CPU for analysis, emphasis added).

g) Appellant further argued, "... *It follows that Ballagh does not describe generating a first output that marks data as valid or invalid, a second output that indicates the first sample of each frame, and a third output that indicates when the model can accept data*" (Brief, page 10, emphasis added).

Examiner respectfully disagrees with Appellant's arguments.

Regarding the reconfigurable filter (FIG.3A) internal/embedded in the FPGA 202 in FIG.2, Ballagh discloses,



"Briefly, the filter operates as follows. When the filter is not being reloaded, input values drive the xn port and filter output values drive the yn port. Filter reloading is initiated with a pulse on the load port, load. During reload, the rfd port outputs zeros to indicate the filter is busy. Following the load pulse, new coefficients are written to the coef port. Asserting coef_we identifies the current value on the coef port as valid. After all coefficients are written, the filter comes back online some number of cycles later and resumes processing data. The filter signals that coefficient reloading is complete by asserting the rfd signal." (col.5: 54-65, emphasis added).

Ballagh discloses:

the FPGA also generating a first output that marks data as valid or invalid (e.g., FIG. 3A, col.5: 61-62 "Asserting coef_we identifies the current value on the coef port as valid"),

a second output that indicates the first sample of each frame (e.g., FIG. 3A, col.5: 54-57, "When the filter is not being reloaded, input values drive the xn port and filter output values drive the yn port", i.e., filter output values including the first output values of each reload drives/indicated by the "yn port"; col.5: 29-38, filter output values as frames), *and*

a third output that indicates when the model can accept data (e.g., FIG. 3A, col.5: 57-58, "During reload, the rfd port outputs zeros to indicate the filter is busy", i.e., if not zeros at the "rfd port", the filter ("model as claimed") can accept data).

h) Appellant further argued, "*In the Response to Arguments on pages 4-5, the final office action asserts that a simulation is performed during the modeling phase of the FPGA (page 4). It then asserts that the yet-to-be-designed FPGA is somehow used to run a portion of the simulation. However, Ballagh is quite clear that the modeling phase is performed by the system of Figure 1. Ballagh is also quite clear that the system of Figure 1 is used to design the circuit 200 of Figure 2, including the FPGA 202. This point was raised in the previous response. The final office action still does not respond directly to this point*." (Brief, page 10).

As an initial matter, in response to Appellant's argument that the references fail to show certain features of Appellant's invention, it is noted that the features upon which Appellant relies (i.e., *yet-to-be-designed FPGA* or *completely-designed* FPGA, emphasis added) are not recited in the rejected claim(s). Although the claims are interpreted in light of the specification, limitations from the specification are not read into the claims. See *In re Van Geuns*, 988 F.2d 1181, 26 USPQ2d 1057 (Fed. Cir. 1993).

After reviewing the Arguments/Remarks filed October 13, 2009 (pp. 4-5), examiner notes that the raised point above was not actually raised in said Arguments/Remarks at all. Pages 4-5 mainly listed the whole claim 35 and discussed about that claim.

Furthermore, the plain language of the claim merely requires *generating input signals and sending signals from a CPU to a FPGA and generating outputs from the FPGA to the CPU* (emphasis added).

The claim language of the claim does not exclude said numerical simulation is performed during a modeling phase of the FPGA (i.e., ***numerical simulation may be***

performed during <u>any</u> phase) and also <u>does not exclude</u> any particular system performs the modeling phase (i.e., ***<u>any</u> system can performs said numerical simulation***).

Accordingly, Ballagh teaches all features of claim 31.

II. Rejection of claims 32-37 under USC 103(a) as being unpatentable over Ballagh in view of Admitted Prior Art (Brief, pp. 11-13)

<u>Claims 32-33 and 35-36:</u> (pp. 11-12)

As an initial matter, examiner notes that Admitted Prior Art (APA) has been applied to reject the particular claimed limitations "*the input signals include sine wave functions representing real and imaginary inputs; and wherein the model includes a FFT.*" Other limitations such as "offloading", "numerical simulation" have been taken care of by the primary reference Ballagh.

Ballagh discloses using <u>a system-level simulation environment 110, such as Simulink</u> (col.4: 47-64, emphasis added), but does not explicitly disclose *wherein the input signals include sine wave functions representing real and imaginary inputs; and wherein the model includes a FFT.*

However, in an analogous art, APA further discloses <u>also using the same software product Simulink</u> (page 2: 20-23, emphasis added) *and the input signals include sine wave functions representing real and imaginary inputs; and wherein the model includes a FFT* (e.g., page 2: 4-23).

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine APA's teaching into Ballagh's teaching. One would have been motivated to do so to transform a digitized waveform in the time domain into a digital representation in the frequency domain by using Simulink simulation software product as suggested by APA (e.g., page 2: 5-9 and 20-23), and <u>not</u> just with a mere "conclusory statement" as Appellant asserted.

Claims 32 and 37: (pp. 12-13)

As an initial matter, examiner notes that Admitted Prior Art (APA) has been applied to reject the particular claimed limitations "*the CPU performs a numerical simulation of a radar system.*"

Ballagh discloses using a system-level simulation environment 110, such as Simulink (col.4: 47-64, emphasis added), but does not explicitly disclose *wherein the CPU performs a numerical simulation of a radar system.*

However, in an analogous art, APA further discloses also using the same software product Simulink (page 2: 20-23, emphasis added) *and wherein the CPU performs a numerical simulation of a radar system* (e.g., page 2: 4-15).

It would have been obvious to a person having ordinary skill in the art at the time the invention was made to combine APA's teaching into Ballagh's teaching. One would have been motivated to do so to transform a digitized waveform in the time domain into a digital representation in the frequency domain by using Simulink simulation software product as suggested by APA (e.g., page 2: 5-9 and 20-23), and not just with a mere "conclusory statement" as Appellant asserted.

(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejection should be sustained.

Respectfully submitted,

/Thuy Dao/

Examiner, Art Unit 2192

Conferees:

/Tuan Q. Dam/
Tuan Q. Dam
Supervisory Patent Examiner, Art Unit 2192

/Lewis A. Bullock, Jr./

Lewis A. Bullock, Jr.

Supervisory Patent Examiner, Art Unit 2193

Attachment 1 – page 1 of 3

US005963731A

United States Patent [19]

Sagawa et al.

[11] Patent Number: 5,963,731

[45] Date of Patent: Oct. 5, 1999

[54] **METHOD OF ASSISTING EXECUTION OF PLURAL SIMULATION PROGRAMS FOR COUPLED SIMULATION**

[75] Inventors: **Nobutoshi Sagawa**, Koganei; **Mikio Nagasawa**, Kodaira; **Shigeo Ihara**, Tokorozawa; **Katsuro Kikuchi**, Hachioji; **Masahiko Hirao**, Kawagoe; **Kirin Ka**, Hachioji; **Satoshi Itoh**, Kodaira; **Yoshio Suzuki**, Kokubunji, all of Japan

[73] Assignee: **Hitachi, Ltd.**, Tokyo, Japan

[21] Appl. No.: **08/773,773**

[22] Filed: **Dec. 24, 1996**

[30] **Foreign Application Priority Data**

Dec. 28, 1995 [JP] Japan 7-338430
Jan. 5, 1996 [JP] Japan 8-145587
Sep. 30, 1996 [JP] Japan 8-258268

[51] **Int. Cl.**[6] **G06F 15/163**; G06F 13/00

[52] **U.S. Cl.** **395/500.27**; 395/500.33; 709/308

[58] **Field of Search** 364/578, 512; 395/500

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,814,612 4/1989 Bruce et al. 364/578
5,386,566 1/1995 Hamanaka et al. 395/700
5,557,774 9/1996 Shimabukuro et al. 395/500
5,701,480 12/1997 James et al. 395/500
5,737,628 4/1998 Rouselle et al. 395/800.13
5,850,345 12/1998 Sue 364/578

FOREIGN PATENT DOCUMENTS

4-336369 11/1992 Japan G06F 15/16

OTHER PUBLICATIONS

"The Mathematics of Public-Key Cryptography" by Martin E. Hellman vol. 241 No. 2 (1979) pp. 130-139.

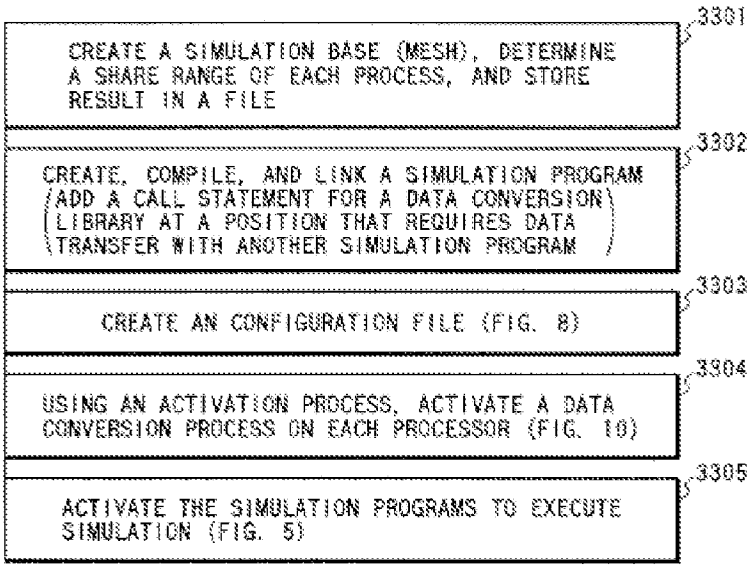
"Cryptography and Computer Privacy", by Horst Feistel vol. 228 No. 5 (May 1973) pp. 15-23

Primary Examiner—Kevin J. Teska
Assistant Examiner—Lonnie A. Knox
Attorney, Agent, or Firm—Antonelli, Terry, Stout & Kraus, LLP

[57] **ABSTRACT**

Each of a plurality of simulation programs is linked with a data conversion library and is executed as a simulation process. A data conversion process is executed in correspondence with each simulation process. In exchanging data resulted from simulation by the simulation process of one of the simulation programs with simulation processes of the other simulation programs, the data conversion process provided for a sending simulation process determines a receiving simulation process to which the data is to be sent, and sends the data to the data conversion process corresponding to the receiving simulation process. The data conversion process for the receiving simulation process performs data conversion for absorbing difference between the base of the sending simulation process and the base of the receiving simulation process, and transfers the data after the conversion to the receiving simulation process.

43 Claims, 31 Drawing Sheets



Attachment 1 – page 2 of 3

METHOD OF ASSISTING EXECUTION OF PLURAL SIMULATION PROGRAMS FOR COUPLED SIMULATION

BACKGROUND OF THE INVENTION

The present invention relates to a method of assisting execution of a plurality of numerical simulation programs for simulating physical phenomenon on a computer so that the programs are executed in cooperation with each other.

Recently, a parallel computer system has become commercially available in which a plurality of processing units (hereinafter also referred to as PUs) are interconnected by way of a high-speed network to be operated simultaneously for enhanced throughput. A region to be simulated is appropriately divided into partial regions, in such a way that each PU has one partial region assigned, thereby effectively exploiting the most of the high processing speed of the parallel computer system.

Parallel computer systems are largely classified into two groups by the method of accessing the memory space from each PU. One group is of shared memory type in which all PUs can access the entire memory space. The other group is of distributed memory type in which each PU can access only a memory space accompanied therewith. Since numerical simulation aims principally at solving large-scale problems by raising computational speeds, use of a parallel computer system of a distributed-memory type is under study.

In a parallel computer system in which a region to be simulated is divided into plural portions and those portions are allocated to a plurality of Pus, exchange of data between the PUs is required during simulation. The data to be exchanged includes results of computation of a physical quantity and the like. For example, a computed value of a physical quantity at a grid point located on the border of a partial region is used for computing a value of the physical quantity at a grid point near the boundary in an adjacent partial region. With a distributed-memory parallel computer system, transfer of data between the PUs must all be described in a program. Data is transferred between the PUs in terms of a message. A program for a parallel computer system must explicitly describe an instruction such that data generated by own PU is transmitted to another PU on generation of the data, in case the data is one required by the another PU, and data held by another PU is received by own PU at a timing when own PU actually uses the data, in case the data is one required by own PU. Many parallel computer systems prepare a group of functions each called a communications library (or subroutines) for supporting the transfer of messages between the PUs. The communication can be described as a function call from C or FORTRAN programs for example. Several communications libraries are implemented across different parallel computer system hardware to provide communications environment as de facto standard. Examples are the PVM (Parallel Virtual Machine) developed by Oak Ridge National Laboratory of US and the MPI (Message Passing Interface) that is under standardization involving many organizations. The parallel programs with calls to these communications libraries are high in possibility (portability) that they can be recompiled on different parallel computer systems for operation.

In executing a simulation program by a plurality of PUs, a plurality of processes for performing simulation of partial regions are executed by the plurality of PUs. In the PVM, a process being executed on each PU is provided with process identification data (PID) that is unique in the PVM. Each of

messages transferred between PUs is composed of main data, the PID, and a message ID (MID) that the user can assign to the message. Therefore, creating parallel programs each of which executes communications based on the PID and MID determined appropriately by the user enables transfer messages between PUs without interference.

A communications library such as the PVM can be implemented on a plurality of workstations interconnected by a network, in addition to parallel computer systems, thereby providing a virtual parallel computer system environment. In operating communications libraries on multiple, different hardware systems, internal data representation may differ from one hardware system to another. That is, if a plurality of workstations interconnected by a network use different central processing units (CPUs) and operating systems (OSs), data types such as integer number and real number in a same program may differ in internal bit representation between the workstations. Therefore, to allow the user of a communications library to make communication without being aware of the details of data representation, the communications libraries such as the PVM provide a capability of automatically converting data representations inside the libraries as required. Providing a PVM process with an appropriate data converting capability allows a message coming from a user process to be converted appropriately and the resultant message to be sent to a destination process. In such a constitution, the differences between OSs and between CPUs are absorbed by the PVM processes, so the user programs are highly independent of the OSs and CPUs.

It should be noted that, when a simulation region of one simulation program is divided into partial simulation regions and simulation processings for them are executed by a plurality of processors concurrently, the mesh density of each partial region may differ from that of an adjacent partial region. According to Japanese Laid-Open Patent Publication No. 4-336369, a technology is disclosed which reflect physical quantities calculated at discrete points on the boundary between of a partial region upon the simulation of an adjacent partial region. That is, a circuit is provided for converting the calculated physical quantities to values at the discrete points on the boundary, belonging to the adjacent partial region and data obtained by this circuit is transferred to an adjacent processor. Actually, the number of boundaries and contour data are entered from both input and output units to create a conversion matrix by the arithmetic unit and the above-mentioned conversion is performed based on the conversion matrix.

Numerical simulation aims at obtaining the distribution of a physical quantity concerned. Numerical simulation is largely divided into two groups by method of representation physical quantity. One is simulations of continuous system and the other is simulations of particle system.

In continuous-system simulation, a physical quantity to be obtained is expressed by f and a space is expressed by a coordinate (x, y) (or (x, y, z) for a three-dimensional space) and the problem comes to solve an equation that f(x, y) satisfies. For example, Maxwell's equations for electromagnetic field description and Navier-Stokes equations for fluid field description are known as the equations of this type. Normally, these equations take a form of partial differential equation; therefore, except for very simple cases, no solution can be obtained analytically. Consequently, an approximate solution is numerically obtained for each problem by means of a computer. To numerically express the distribution of f(x, y) in the computer, a concept of functional discretization is introduced. Discretization denotes an operation for approximately representing, in the finite number of numerical values, an originally continuous function.

Attachment 1 – page 3 of 3

graph **103**). The acquisition of the computer usage status **105** is performed by a resource monitoring demon **104** resident on one of the computers. The resource monitoring demon uses an OS system call to acquire and hold the computer usage status. In addition, the resource monitoring demon monitors the execution state of the above-mentioned coupled program and automatically register the program resource database (**105**).

Referring to FIG. **31**, a file in which paths to the executable forms of the element programs for the coupled program to be input are described and a file in which IP addresses (or host names) for identifying computers to be used are passed to a loader program (hereinafter referred to as a loader system) prepared by the user (block **201**).

The loader system checks the computers described in the file for availability and extracts the available computers (block **202**). An available computer herein denotes a computer that is connected both physically and logically to a computer made a host computer by the user by inputting a program input command. An unavailable computer denotes a computer having no effective path with the host computer. A computer is made unavailable when the same is not networked, a network coprocessor (a processor unit dedicated to checking destinations of data flowing through the network and transferring the data) if any of the computer is failing, the computer is in the halt state, or the network to which the computer is connected is failing, by way of example.

Then, the resource consumption quantity of each element program is acquired. The resource consumption quantity includes a load quantity, a memory request quantity, an average file I/O quantity (hereinafter referred to as a file I/O quantity), and an average transfer data quantity between element programs (hereinafter referred to as a transfer data quantity). To be specific, the load quantity is the total number of four basic arithmetic operations that appear in the program. Triangular functions and arithmetical functions such as square roots are converted to the number of four basic arithmetic operations. The memory request quantity denotes a memory quantity to be consumed by the program. This quantity is equivalent to the total quantity of codes, stacks, and data of the program, because the data occupy most of this quantity, the quantity of the data to be processed by the program is used for the memory request quantity for simplicity. The average file I/O quantity indicates an average quantity per every file input/output or disk input/output that appears in the program. The transfer data quantity denotes an average quantity per every data input/output of message exchange between element programs. Information about these resource consumption quantities is registered in the program resource database **102** in advance and therefore acquired from this database. For a new program, the resource consumption quantity information is not yet registered in the program resource database **102**. In this case, if any of the element programs constituting the coupled program has the resource consumption quantity information registered in the database, appropriate values are set based on that information. If none of the element programs has the resource consumption quantity information, appropriate values are set based on the resource consumption quantity information of another element program in the program resource database **102**. For these values, an average value of the registered element programs is set for example.

If none of the resource consumption quantity information is registered, a uniform resource consumption quantity is assumed. Further, if no resource consumption quantity information is registered in the program resource database **102**, the resource consumption quantity is collected by a resource information automatic acquisition feature during the execution of the coupled program to be reflected on the program resource database **102**. This automatic acquisition feature will be described later. Also, the user can define the resource consumption quantity information. In this case, it is possible to give preference or a weighted addition between the information stored in the database **102** and the information defined by the user for the resource consumption quantity information. In the present embodiment, the resource consumption quantity information defined by the user is preferred by way of example. In addition, when the resource consumption quantity is unknown, it may be acquired before, not during execution, of the coupled program by interactively inquiring the user (block **203**).

Then, the resource quantity of each computer is obtained. The resource quantity includes a CPU power, a main memory quantity, a disk I/O power, and a network I/O power. To be specific, the CPU power denotes an arithmetic capability of the processor, or an arithmetic operation quantity executable in a unit time. In the simulation program used in the present embodiment, floating-point operations are mainly performed, so that FLOPS (Floating-point Operations Per Second) is used for the unit for the CPU power. The main memory quantity denotes a main memory quantity held by the processor. The disk I/O power denotes a data transfer rate at the time when the program performs a disk I/O operation on that processor. For this data, a peak value of the disk I/O performance is used. The network I/O power denotes a data transfer rate of the network interconnecting the processors. For this data, a peak value of network transfer rate performance is used. The information of these resource quantities is stored in the computer resource database **101** and therefore is referenced from this database. If no resource quantity information is registered as with a new computer, appropriate values are set based on the registered information if any of other computers. For these values, an average value of the registered computers is set for example. If none of the resource consumption quantity information of the other computers is registered, a uniform resource consumption quantity is assumed. Also, the user can define the resource consumption quantity information. In this case, it is possible to give preference or a weighted addition between the information stored in the database **101** and the information defined by the user for the resource consumption quantity information. In the present embodiment, the resource consumption quantity information defined by the user is preferred by way of example (**204**).

Then, the current usage states of each computer is obtained from the resource monitoring demon. The resource usage status indicates the usage status of the computer resources such as CPU, memory, disk I/O, and network I/O (**205**). To be specific, a CPU usage ratio, a free memory quantity of main memory, an average disk I/O quantity, and an average network I/O quantity. The average disk I/O quantity denotes a disk I/O quantity per second and the average network I/O quantity denotes a network I/O quantity per second.

Lastly, the above-mentioned coupled program is allocated to the computers based on the above-mentioned resource information to actually load and execute the coupled program (**206**).

The following describes in detail the constitutions of the computer resource database **101** and the program resource database **102**, the resource monitoring demon, the program allocation, and the resource information automatic acquisition feature.